COBB

Inside MICROSOFT ACCESS

May 1994 • Vol. 2 No. 5

Tips & techniques for Microsoft Access • Windows

Creating help labels to pop up descriptions of the current field

Design Tip

By Phoebe Spinrad

Phoebe Spinrad is a professor of Renaissance and medieval literature at Ohio State University in Columbus, Ohio, and offers Access consulting services in the area. You may reach her at (614) 258-5433 or fax questions to (614) 258-6225.

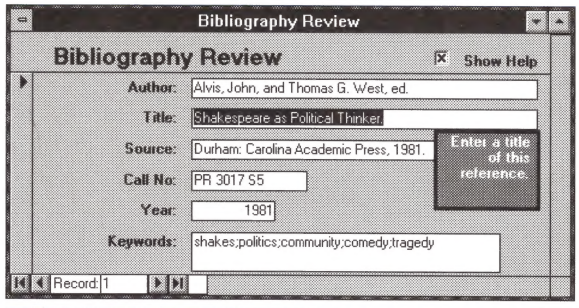
When you're creating Access applications for other people, you often want to instruct the users how to fill in the fields on the forms. To do so, you might want your form to display pop-up message boxes that provide help messages as the user enters the fields. However, if you display a message box, the user must close the message box before continuing on to another field. You ideally want a message to appear and then disappear by itself so the user doesn't need to click an OK button before leaving the control.

In this article, we'll show you how to create a "help label" that pops up as you enter a field and then vanishes as you leave it. As a user tabs through the form, explanations will appear for every field. For example, Figure A shows a form that displays the help message *Enter a title of this reference* as a user enters the Title text box. Each text box on this form has a similar pop-up message that appears when the user enters the field. We'll also show you how to enable a user to turn off the messages.

The technique

To create a help label, you place a label control next to the field for which you want to display the help message. You then enter the help text you want to display and adjust the size and shape of the label control. Next, you set the help control's Visible property to *No*. Remember, you don't want all the help labels on the screen at once. You want them to be invisible until the user enters a control. You then want to set the Visible property to *Yes*.

Figure A



This form pops up help messages when you enter the fields.

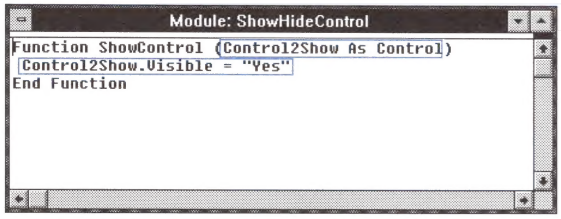
IN THIS ISSUE

You don't need to worry if the help labels overlap other controls as long as the help labels reside on top when visible. There's one exception to this guideline: You don't want the help label to overlap the control for which it provides help. The current control always rises to the top. If the help label overlaps its associated control, the control will obscure the help message.

You might also want to open the palette and format the color or border thickness of the help labels. That way, the user can distinguish the help messages from other labels and text boxes on the form.

Also, you must assign the help labels control names as you create them. The Access Basic functions we'll show you refer to the controls' names in order to set their Visible property.

Figure B



Complete the ShowControl() function as shown here.

The ShowControl() and HideControl() functions

We'll now show you how to create the Access Basic functions you'll use to set the Visible property. You'll create the ShowControl() function to make a help label visible and assign that function to the label's On Enter property. You'll create the HideControl() function to make a help label invisible and assign it to the On Exit property.

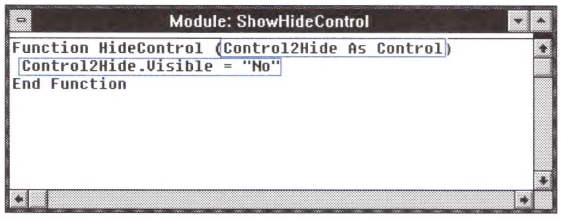
Start by clicking the Module button in the Database window and then clicking the New button. When the Module window appears, pull down the Edit menu and select the New Procedure... command. In the New Procedure dialog box, enter ShowControl and click OK.

After Access inserts the new function definition in the Module window, complete the function as shown in Figure B. We've placed in boxes the statements you'll need to add.

Next, issue the New Procedure... command again and enter *HideControl* into the New Procedure dialog box. When the new function name appears, complete the function as shown in Figure C, inserting the statements in the boxes.

Both functions accept the control you want to show or hide as a function argu-

Figure C



Complete the HideControl() function as shown here.

Inside MICROSOFT ACCESS™

Inside Microsoft Access (ISSN 1067-8204) is published monthly by The Cobb Group.

Prices
Domestic ..... $59/yr. ($7.00 each)
Outside US ..... $79/yr. ($8.50 each)

Phone
Toll free ..... (800) 223-8720
Local ..... (502) 491-1900
Customer Relations Fax ..... (502) 491-8050
Editorial Department Fax ..... (502) 491-4200

Address
You may address tips, special requests, and other correspondence to
The Editor, *Inside Microsoft Access*
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

For subscriptions, fulfillment questions, and requests for bulk orders, address your letters to
Customer Relations
9420 Bunsen Parkway, Suite 300
Louisville, KY 40220

Advertising
For information about advertising in Cobb Group journals, contact Tracee Bell Troutt at (800) 223-8720, extension 430.

Postmaster
Second class postage is pending in Louisville, KY. Send address changes to
Inside Microsoft Access
P.O. Box 35160
Louisville, KY 40232

Authorized Canada Post International Publications Mail (Canadian Distribution) Sales Agreement #XXXXXX CANADA GST #123669673. Send returns to Canadian Direct Mailing Sys. Ltd., 920 Mercer Street, Windsor, Ontario, N9A 7C2. Printed in the USA.

Copyright
Copyright © 1994, The Cobb Group. All rights reserved. *Inside Microsoft Access* is an independently produced publication of The Cobb Group. The Cobb Group reserves the right, with respect to submissions, to revise, republish, and authorize its readers to use the tips submitted for both personal and commercial use.

The Cobb Group, its logo, and the Satisfaction Guaranteed statement and seal are registered trademarks of The Cobb Group. *Inside Microsoft Access* is a trademark of The Cobb Group. Paradox is a registered trademark of Borland International. dBASE III and dBASE III PLUS are registered trademarks of Ashton-Tate, a Borland International company. Microsoft, MS-DOS, and Access are registered trademarks of Microsoft Corporation. Microsoft Windows and Word for Windows are trademarks of Microsoft Corporation.

Staff
Editor-in-Chief ..... David Brown
Editors ..... Meredith Little
Polly Blakemore
Elizabeth Welch
Production Artists ..... Maureen Spencer
Margueriete Stith
Design ..... Karl Feige
Managing Editor ..... Elayne Noltemeyer
Circulation Manager ..... Marjorie Glassman
Editorial Director ..... Jeff Yocom
Publishers ..... Mark Crane
Jon Pyles

Back Issues
To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $7 each, $8.50 outside the US. You can pay with MasterCard, VISA, Discover, or American Express, or we can bill you. Please identify the issue you want by the month and year it was published. Customer Relations can also provide you with an issue-by-issue listing of all articles that have appeared in *Inside Microsoft Access*.

ment. Note that the functions don't accept the string name of the control. You must declare the argument as a Control data type. The single line of each function then sets that control's Visible property. The function ShowControl() sets Visible to *Yes*; HideControl() sets the property to *No*.

Once you've finished entering the functions, pull down the Run menu and click Compile All. Then, save the module with the File menu's Save command, entering *ShowHideControl* in the Save As dialog box. Finally, close the Module window.

An example

Now that you've created the ShowControl() and HideControl() functions, you're ready to build a form that displays the help messages. Suppose you need to create a data entry form for the Biblio table shown in Figure D. This table stores bibliographic data for a list of the references you often use in research projects. The Author and Title fields compose the table's key.

After you've created this table in your test database, highlight the table name in the Database window and click the New Form button (▤) in the tool bar. In the New Form dialog box, click the FormWizards button. In the next dialog box, highlight the Single-Column entry and click OK. The Single-Column form wizard will then present several dialog boxes that ask you to supply information about the new form.

In the first wizard dialog box, double-click the fields Author, Title, Source, Call No, Year, and Keywords. Then, click the Fast Forward button (▶▶) in the lower-right corner of the dialog box. In the final dialog box, click Design. The wizard will generate a form with the default layout shown in Figure E.

The first field that appears on the form is Author. To create a help label for this field, first select the Label tool (A) in the tool box. Then, place a new label below and along the right side of the Author text box, as shown in Figure F. As we said before, the help label shouldn't overlap the control for which you want to show help. However, it can overlap the neighboring controls.

When you release the mouse button after placing the label, Access will leave the cursor in the label so you can immediately provide the help message. Type *Enter the author for this reference* and press [Enter].

Next, you'll use the palette to format the label. Open the palette by clicking the Palette button (▣) on the tool bar. Then, deselect the two Clear check boxes. By default, both

Figure D

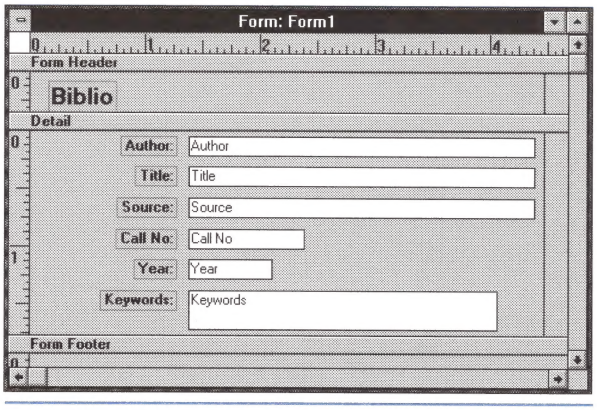
Table: Biblio

Author	Title	Source	Call No	Year	Keywords	
Bawcutt, N. W.	"He Who the Sword of Heaven Will Bear': The Duke Versus Angelo in	Shakespeare Survey 37 (1984):	PR 2888 S52	1984	mform;shakes;law;comedy	0
Bellamy, John	The Tudor Law of Treason.	London: Routledge &	UNK	1979	law; treason	0
Boorman, S. C.	Human Conflict in Shakespeare.	London and New York: Routledge	PR 2989 B66 1987	1987	shakes;comedy;community;law	0
Brownlow, F. W.	'John Shakespeare's Recusancy: New Light on an Old Document.'	Shakespeare Quarterly 40	UNK	1989	shakes;community;law	0
Clark, P., and P. Slack	English Towns in Transition, 1500-1700.	London: Oxford UP, 1976.	HT 133 C52	1976	community;justice;law;police	0
Clark, P., and P. Slack, ed.	Crisis and Order in English Towns, 1500-1700.	London: Routledge &	HT 133 C5	1972	politics;law;community	0
Draper, J.W.	'Dogberry's Due Process of Law.'	JEGP 42 (1943): 563- .	PD 1 J6	1943	shakes;muchado;law;community;police	0
Elton, G. R.	Policy and Police: The Enforcement of the Reformation in the Age of	Cambridge: Cambridge UP,	DA 332 E496	1972	law;politics;religion;police	0
Fitzherbert, Anthony	The New Boke of Justices of the Peas.	London, 1538. STC 10969.	JN 841 P4 F5 1538A	1538	community;justice;law;police	0

Record: 1

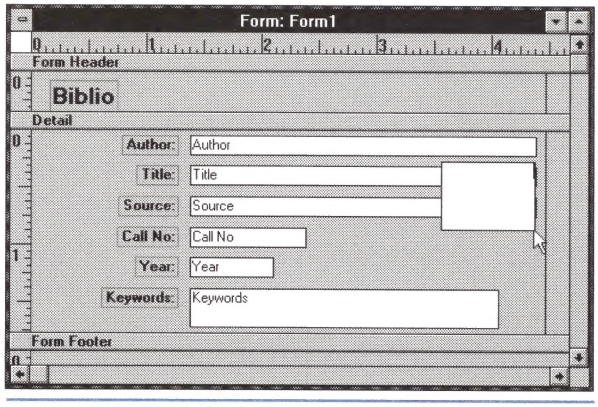
You'll create a data entry form for the Biblio table that displays help controls as you move among the fields.

Figure E



The Single-Column wizard will create this default form, displaying the six fields of the Biblio table.

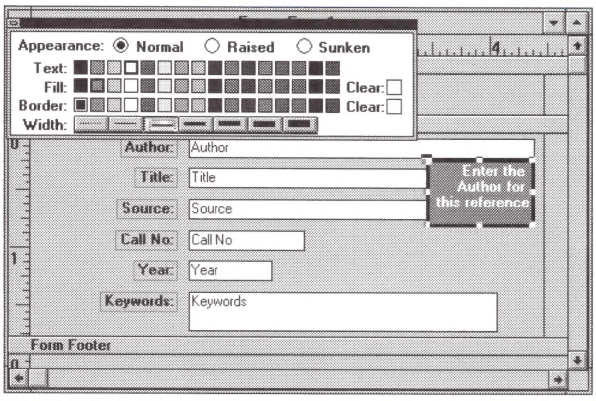
Figure F



You place a help label adjacent to the text box for which you want to display help.

the interior and the border initially will be transparent. Now, set the label's color properties to white text on a dark gray background. To do so, click the white box in the Text row and the dark gray box in the Fill row. To set the border thickness, click the third selection from the left in the Width row. The label should look similar to the one shown in Figure G.

Figure G



Format the help labels like this one.

The label should be on top of all the other controls. However, if you clicked on other controls as you worked through our directions, other controls may reside on top of the help label. The help label must reside above the other controls on the form. So, if you're not certain that the help label is on top, select the label and issue the Bring To Front command from the Layout menu.

Now you need to set a couple of label properties. Open the property sheet by clicking the Properties button (▣) on the tool bar. Then, assign *AuthorHelp* to the Control Name property. The ShowControl() and HideControl() functions will refer to the label by this name. Next, move to the Visible property and enter *No*.

Finally, you must set the On Enter and On Exit properties of the text box that will

Displaying field explanations in the status bar

In the accompanying article, we show you how to place labels on a form that display help messages as you tab through the form. The help labels are perfect for when you're developing a form for users who aren't comfortable with computers. The help messages can be as long as necessary and are immediately apparent to users.

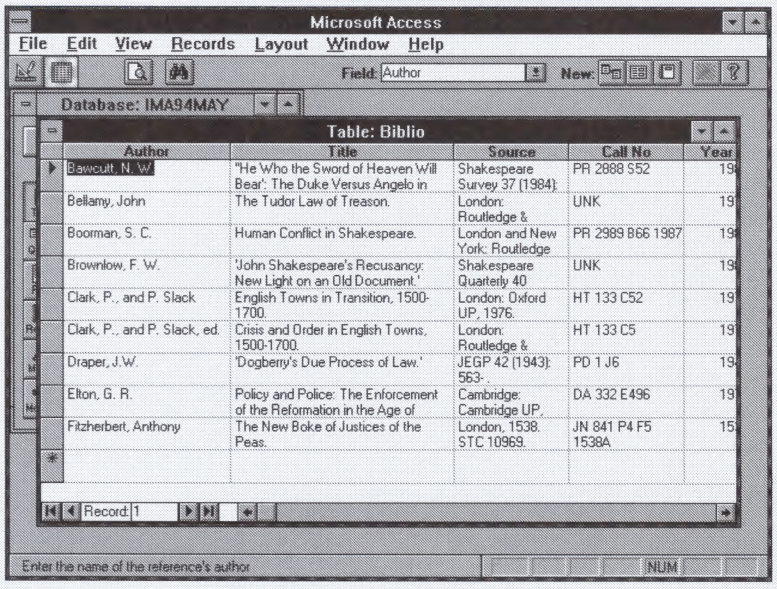
However, if you're designing a form for yourself or for experienced users who will catch on quickly, creating all those help messages might be overkill. Instead, you can display a descriptive message in the Access status bar along the bottom of the Access window. In this sidebar, we'll describe how.

Creating field descriptions on a datasheet

You create field descriptions by opening the table—not the form—in Design View and typing the field messages you want to display in the Description column. You then close the table and save your changes. When you next view the datasheet, Access will display your message in the status area at the bottom of the Access window whenever you enter that field.

You can easily try this technique with the Biblio table we described in the main article. Open the table in Design View, move to the Author field's Description cell, and type *Enter the name of the reference's*

Figure A



Your Description cell entries appear in the status area at the bottom of the Access window.

trigger the help message. To activate the AuthorHelp label, you want to assign your functions the Author text box's On Enter and On Exit properties. To do so, select the Author text box and, in the property sheet, assign

```
=ShowControl([AuthorHelp])
```

to the On Enter property and

```
=HideControl([AuthorHelp])
```

to the On Exit property.

The AuthorHelp label will now appear when you enter the Author text box and disappear when you leave it. Before creating the other help labels, switch to Form View to make sure your first help label works properly. Because the Author text box is the first control in the form's tab order, the message will appear when you open the form. However, it should disappear when you move to the Title control.

Next, return to Design View and place the other controls' help labels. Use the Edit menu's Duplicate command to generate labels with the formatting you set earlier. To do so, select the help label you just created, issue the Duplicate command, and place the duplicate label below the next control. Then, change the Control Name property and update the On Enter and On Exit properties so that they pass the correct control name to the functions. And, of course, change the help message and appropriately position the label next to its control. Repeat these steps for each help label and control. When you've *author*. Then, click the Datasheet View button (▦) on the toolbar and save your changes. When you enter the Author field, the message appears at the bottom of the Access window, as shown in Figure A.

Creating field descriptions on a form

Now when you create a form based on the table, the form will also display your field descriptions. Why? Well, form controls have a property named Status Bar Text in which you can enter field descriptions. Access will use the descriptions you provided in the table's Design View as the Status Bar Text property settings for new controls.

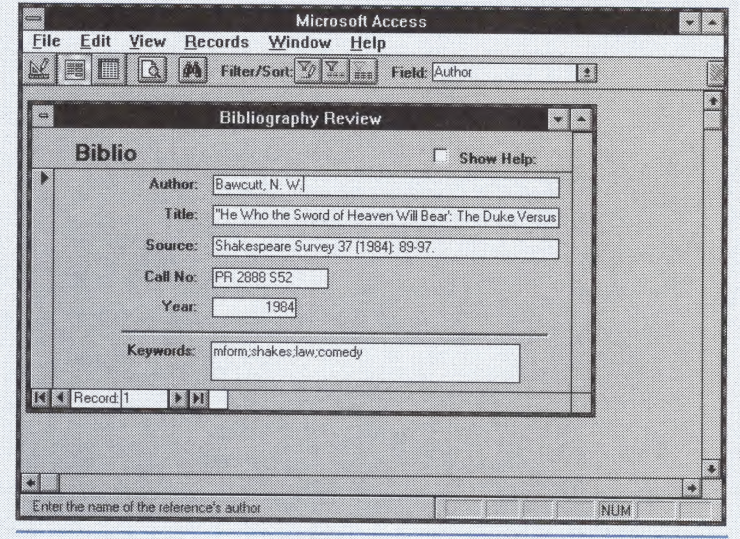
As you might expect, Access will not copy the table's field descriptions to the controls of existing forms. You must open the form in Design View and either enter the descriptions yourself or place the controls again.

Once you enter the field description in the Status Bar Text property, the description will appear in the status bar as you tab to that control on the form. Figure B shows our message in the status bar for the Author field.

Conclusion

The size of the status area constrains the length of your message. Furthermore, the message won't be very obvious as you work with the form. However, experienced users will know to look there for help. The field descriptions often let you provide all the help experienced users will need.

Figure B



After we set the Author control's Status Bar Text property, our field description appears in the status bar when we enter the field.

finished, your form will look a mess—as does our form shown in Figure H. However, only one help message at a time will appear in Form View.

Letting the user turn off the help messages

Although the help labels can be invaluable to a new user, experienced users might get annoyed with the ever-present messages. Therefore, you may want to enable users to turn off the messages. We'll next show you how to place a check box in the form's header that the user can click to choose whether the messages will appear.

If you left the form in Form View, switch back to Design View. Then, select the Check Box tool (☑) from the tool box and place a new check box in the Form Header section.

Next, open the property sheet and assign *Show Help* to the Control Name property. Then, set the Default Value property to 0 if you want to set the default to no messages. If you want the messages to appear by default, set the property to -1. You'll also want to change the caption of the associated label. Select the label to display its properties in the property sheet. Then, change the Caption property to *Show Help*. The Show Help check box will appear on your form, as shown in Figure I.

Of course, the presence of the check box doesn't suppress the help labels. Its value simply gives the go-ahead to the text boxes to display the help labels. You must update all the text boxes' On Enter and On Exit properties so that they examine the Show Help check box before displaying the help labels.

You can do this most easily by calling the ShowControl() and HideControl() functions from IIf() function calls. You can create an IIf() call that executes ShowControl() and HideControl() only if the Show Help box is checked. For example, you change the Author text box's On Enter property to

```
=IIf([Show Help] <> 0, ShowControl([AuthorHelp]))
```

We've printed in light blue the new components of the expression you'll need to enter. You'll have to insert the shaded components of the expressions in all the text boxes' event properties.

Similarly, you must change the text boxes' On Exit property to

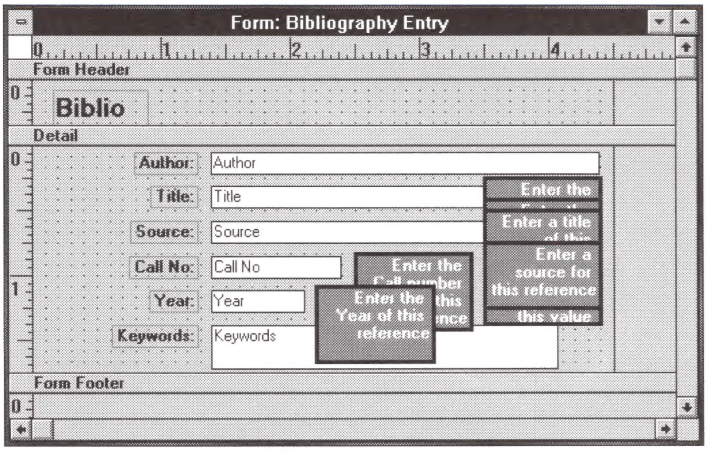
```
=IIf([Show Help] <> 0, HideControl([AuthorHelp]))
```

so that Access will call the HideControl() function only if the Show Help box is checked. Again, we've printed in blue the components you'll need to insert. You must update the On Exit property of all text boxes with the new components.

Conclusion

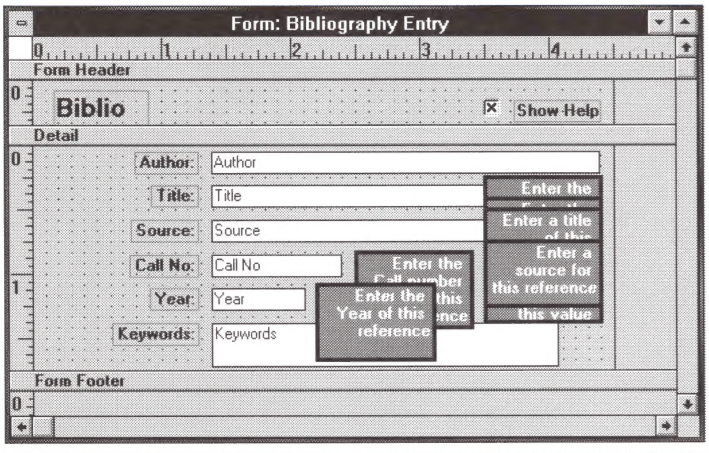
In this article, we showed you how to create help labels on your form that give the user detailed instructions during data entry. By using help labels, you don't need to call distracting message boxes or write separate help files. You hide and show help labels by calling the Access Basic functions ShowControl() and HideControl(), which set the help labels' Visible property. ❖

Figure H



In Design View, you'll see all the help labels at once, but don't worry—Form View will show only one label at a time.

Figure I



You can place a check box in the form header that will let the user turn the messages on or off.

Replacing frequently called macros with generic Access Basic functions

In the article "Creating Help Labels to Pop Up Descriptions of the Current Field" on page 1, we show you how to create two Access Basic functions to turn on and off the Visible property. You may have wondered why we didn't use macros rather than functions to set the controls' Visible property. If you don't have much programming experience, the advantages of using functions may not be obvious. In this article, we'll point out those advantages and outline the situations in which you should choose functions over macros.

Using macros to set the Visible property

You certainly *could* create macros to set and reset the Visible property. A macro would simply run a SetValue action. To make a control visible with the SetValue action, you assign that control's Visible property to the Item argument and the string *Yes* to Expression. To make a control invisible, you assign *No* to the Expression argument. For example, the macro shown in Figure A sets the AuthorHelp control's Visible property to *Yes*.

The macro does a perfectly good job of setting the Visible property. Unfortunately, you must specify the name of the control in the Item argument. As a result, you must create a pair of macros for each control—one macro to turn on the control's help label and another macro to turn it off. If you needed to set only one control's Visible property, this wouldn't be a problem. However, when you're setting the Visible property of all a form's help labels, you might need to create dozens of macros.

Creating functions

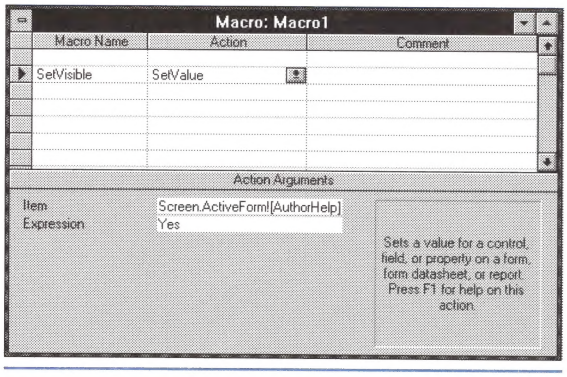
By making the leap to Access Basic, you can escape this duplication of effort. You can create ShowControl() and HideControl() functions that will work for any control by passing the control name as an argument. The help label's control name is the only piece of information that changes. You still need to call the function in each control's On Enter and On Exit properties. However, you can call the same functions for each control and supply the relevant control name as the argument.

An important lesson

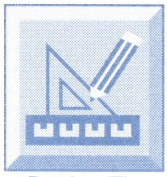
There's an important programming lesson here: Whenever you automate a certain task, you should determine whether the steps of the task will always be the same. You next want to isolate the quantities that will change from time to time. If you're automating a calculation, try to identify the quantities you'll need to provide as input. You can then write the Access Basic statements that will perform the tasks or calculations, and you can define as function arguments the quantities that will vary. Such a function is called generic, which means that the function doesn't care about the environment in which you call it. As long as you pass arguments to the function, it can do the job.

The ShowControl() and HideControl() functions are generic. The only thing that changes is the help label's control name, which you pass as the sole function argument. Notice also that you can use these functions in a variety of situations and for a variety of control types—not just for showing and hiding label controls. You can use these functions whenever you want to hide or show any control that has a Visible property. ❖

Figure A



This macro sets the AuthorHelp control's Visible property to Yes.



Creating a chiseled effect for your form controls

If you've used Access for long, you've probably used the Special Effect property to enhance the appearance of your forms. For instance, by setting a control's Special Effect property to *Sunken*, you create a recessed effect for the control. By setting it to *Raised*, you create an embossed effect.

In this article, we'll show you how to combine these special effects to create controls that have a chiseled border. The borders of a chiseled control will appear recessed, but the interior will remain level with the form's background.

Creating chiseled lines

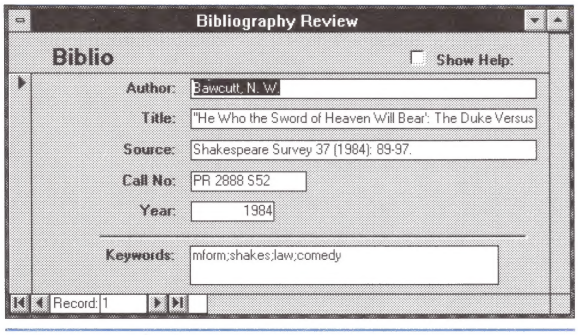
A line is the simplest type of control you can chisel. A chiseled line looks like a thick line with a recessed interior. For example, Figure A shows a form that uses a chiseled line to set off one section.

You don't actually place a line control to achieve this effect. Instead, you place a very narrow rectangle control and apply the sunken special effect to it.

Let's create a chiseled line on a new form. You'll start by opening a test database and clicking the Form button to list the database's forms. Next, click the New button and then click the Blank Form button in the New Form dialog box. When the blank form appears, open the palette by clicking the Palette button (▣) in the tool bar. You must first set the form's background color. Click the Detail section bar to select the section. Then, click the gray box in the palette's Fill row.

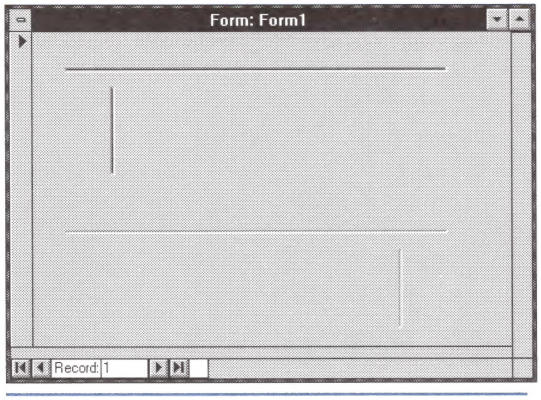
Now, place the thin rectangle control by using the Rectangle tool (▭). Back in the palette, click the Sunken button in the Appearance row. Next, you open the control's property sheet by clicking the Properties button (▤) in the tool bar. Set either the Width or Height property to 0.02. You set the Width property when you want to create a vertical chiseled line, and you set the Height property when you want a horizontal line. The upper half of Figure B shows a line of each orientation. We created the lines in the lower half of the figure using the same method. However, we set the thickness of these lines to 0.01. As you can see, the chiseled effect is more subtle for these lines.

Figure A



This form sets off its last section with a chiseled line.

Figure B



By using rectangle objects and the sunken special effect, you can create chiseled lines.

Creating chiseled rectangles

As you'd expect, creating a chiseled rectangle is more complicated. You actually create two rectangles, one slightly larger than the other. You put the smaller one on top and format it with the raised special effect. You format the larger rectangle on the bottom with the sunken special effect. The combination of the special effects creates the chiseled groove between the rectangles' borders.

To see an example of a chiseled rectangle, open the Print Setup dialog box by issuing the File menu's Print Setup... command. As you can see in Figure C, this dialog box uses chiseled rectangles to create sections of controls.

To create a chiseled rectangle, first return to Design View of your test form. Then, pull down the Layout menu and select Snap To Grid. You'll see how this command comes into play shortly.

Next, select the Rectangle tool from the tool box and place the rectangle to which you want to give the chiseled look, as shown in Figure D. Next, use the palette to match

the rectangle's background color to the form's background color by selecting the gray box in the Fill row. Then, click the Raised radio button in the Appearance row. This rectangle will be the interior control that provides the rise out of the chiseled groove.

You'll next place the exterior rectangle exactly on top of the first rectangle and then enlarge it slightly. Start by pulling down the Edit menu and selecting Duplicate. Access will make a copy of the rectangle, setting it off from the first. Click the rectangle's move handle and drag it on top of the first rectangle. Because you issued the Snap To Grid command before placing the original rectangle, you'll be able to overlap the rectangles easily, as the first rectangle occupies one of the positions on the grid.

To enlarge the second rectangle, you use the property sheet to set the four properties Left, Top, Width, and Height. Click the Properties button (▤) on the tool bar to open the property sheet. The properties will match the original rectangle's property values. Decrease the Left and Top properties by 0.01 and increase the Width and Height properties by 0.02. By making these changes, you extend all four sides by 0.01 inches. It may not sound like much, but it's enough to create the chiseled effect.

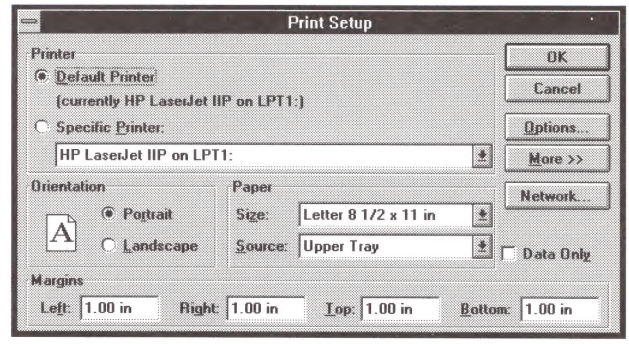
To complete the effect, you place the second rectangle behind the first by pulling down the Layout menu and selecting Send To Back. You then return to the palette and change the Appearance selection to Sunken. To see the result, click the Form View button (▤) in the tool bar. Your rectangle will resemble the one shown in Figure E.

Labeling your chiseled rectangles

As we mentioned, the Print Setup dialog box in Figure C uses chiseled rectangles to partition the dialog box into sections. Also, it uses labels to provide section names. You can create similar labels for your chiseled rectangles. To do so, first select the Label tool (A) in the tool box and then place the new label control on the left side of the rectangle's top border. Access will leave the cursor in the new label. Type the label text *Sample* and press [Enter]. The label should overlap the chiseled border, as shown in Figure F.

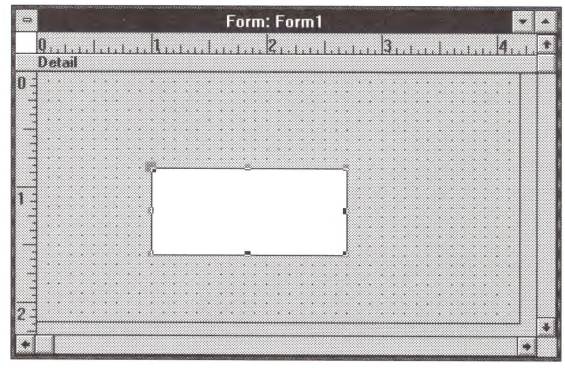
Next, return to the palette to format the new label. Start by deselecting the Clear box in the Fill row. You want the label to

Figure C



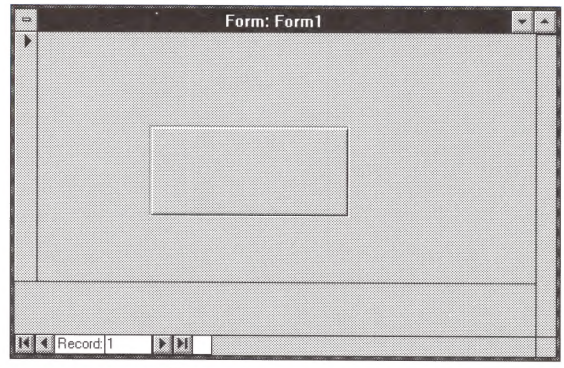
We'll show you how to give your forms the chiseled effect shown here.

Figure D



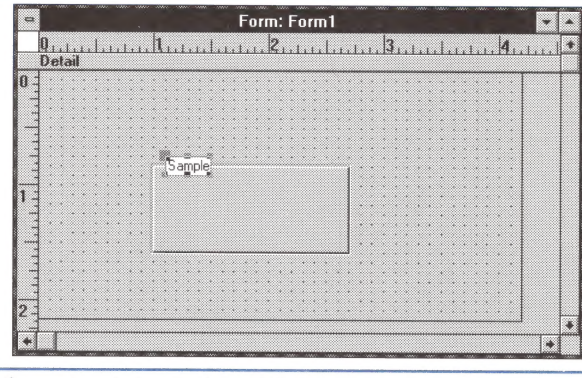
You first create a rectangle control.

Figure E



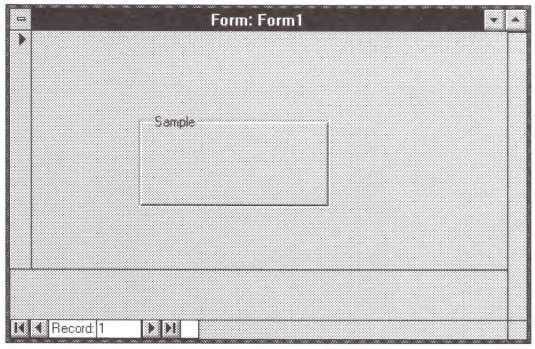
Your completed rectangle will have a chiseled border.

Figure F



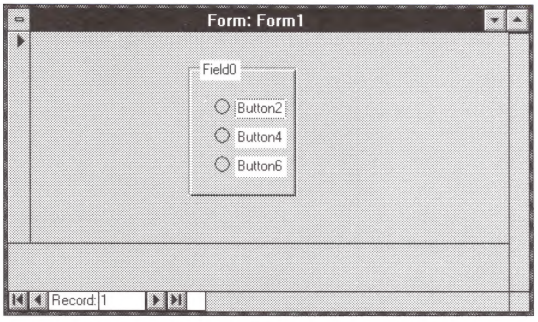
You label the rectangle by placing a label control on the rectangle's top border.

Figure G



You complete the rectangle's label by removing its Clear property and painting it gray.

Figure H



We created a chiseled option group by using the same method as we used for rectangles.

obscure the border. Next, click the gray box in the Fill row so the label will blend into the background. Figure G shows the result.

Creating the chiseled effect for other controls

We've walked you through the steps for creating chiseled rectangle controls; you can also create chiseled text boxes, list boxes, option groups, and so on. You place the control first, formatting it with the form's background color and the sunken appearance. Then, create the rectangle that will complete the chiseled effect.

Of course, you can't create the rectangle by using the Duplicate menu command. Instead, you must use the Rectangle tool to place the rectangle and assign its color and appearance with the palette.

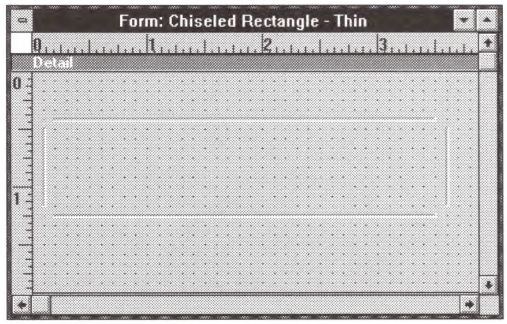
As an example, Figure H shows a chiseled option group. We won't walk you through this example because the steps are the same. You just start with an option group control and then create the sunken rectangle that provides the chiseled effect.

Piecing together chiseled lines to soften the groove of a chiseled rectangle

If you compare the chiseled rectangles in the Print Setup dialog box with the one you just created, you'll notice that the grooves are deeper in your rectangle. However, the two rectangles are as close to each other as possible. You can't soften the groove by reducing the space between the rectangles. If you want your chiseled rectangles to look like the ones in the Print Setup dialog box, you must piece together four chiseled lines—one for each border.

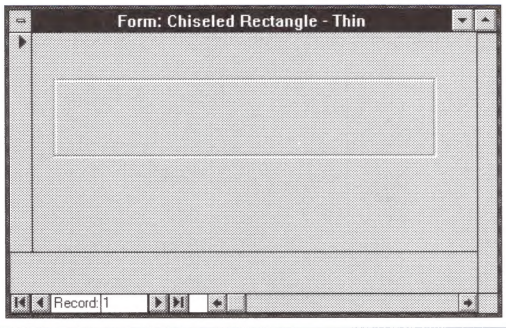
Start by creating the thin chiseled lines as we described on page 8. In other words, create four thin rectangle controls. For the two vertical lines, set their Width property to 0.01. To create the two horizontal lines, set their Height property to 0.01. Then, open the palette and check the Sunken box in the palette's Appearance row. Figure I shows the four component lines. Figure J shows the final chiseled rectangle after you drag the four lines so that their ends meet.

Figure I



You can build a softly chiseled rectangle by first creating chiseled lines.

Figure J



You connect the lines' ends to create the rectangle.

Conclusion

In this article, we showed you how to create a chiseled effect by using the sunken and raised special effects. In summary, you create a chiseled line by placing a very narrow rectangle and applying the sunken effect. You create a chiseled look for a rectangle or any of the other rectangular controls by applying the raised effect to the control and then creating a slightly larger rectangle control formatted with the sunken effect. In both cases, the chiseled groove appears where the sunken and raised effects meet. ❖

Customizing the default properties of form and report controls

Have you ever found yourself reformatting control after control with the same properties? For example, suppose you want to format all text boxes on a report as right-aligned. After placing every text box, you must change the Text Align property from the default value *General* to *Right*. If your report includes a lot of fields, updating all the properties can take a long time.

Fortunately, you can change a control's default properties so that controls you create will have your custom default properties. In this article, we'll describe how you set default properties. We'll then demonstrate how setting your own defaults can simplify form and report design.

Defining custom default properties

To define default properties for a control, you open the property sheet and then select the control's tool in the tool box. The property sheet will list the properties you can set for that control. You assign your default property settings simply by setting the values in the property sheet.

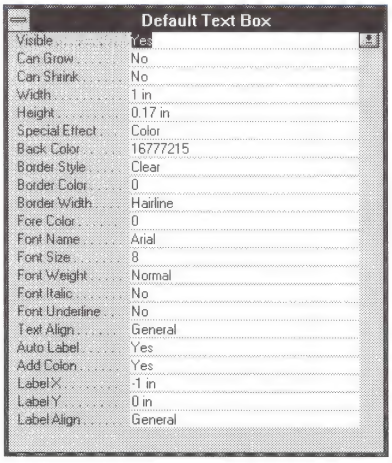
Let's return to our example: You're about to create a report on which you'll place lots of text box controls, and you want all the text boxes to print their values right-aligned. Before placing any fields, open the tool box and select the text box tool (ab|). Then, open the property sheet by clicking the Properties button (▤) on the tool bar. The property sheet will list the default properties that new text boxes will have, as shown in Figure A. Find the Text Align property and set it to *Right*. Now all text boxes you create will right-align their contents.

Notes

Access saves the default properties only for the current form. For instance, when you save the report we just described, the report object will remember your default settings. If you close the report and then open it again in Design View, any text boxes you create will be right-aligned. However, the controls for reports you create will have the Access default properties.

You may find that being able to define a control's default properties will reduce your use of wizards. After all, the purpose of wizards is to quickly place all fields with a common formatting. If you don't like the wizard's formatting, you must reformat all the controls anyway. By setting the default properties of controls before you start placing them, you can format them just as well as the wizard. Furthermore, when you place controls yourself, you can immediately put them where they belong. When you use a wizard, you almost always need to fiddle with the placement of the controls. ❖

Figure A



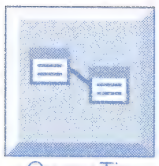
This property sheet lists the default properties for text box controls.

Subscribe to Inside Microsoft Access Resource Disk!

Do you wish that you could experiment with the forms, reports, tables, macros, modules, and queries we regularly feature in *Inside Microsoft Access* but don't have the time or patience to create them? If so, you may want to subscribe to Inside Microsoft Access Resource Disk. Once you subscribe, we'll send you a disk loaded with all the tips featured in that month's *Inside Microsoft Access*. (See the articles marked with the disk icon.)

A six-month subscription to Inside Microsoft Access Resource Disk costs $29. A full one-year subscription is $49. If you don't want to subscribe but would like the forms, reports, tables, macros, modules, and queries in a particular issue of *Inside Microsoft Access*, you can purchase a single disk for only $9.95.

To subscribe or order a specific month's disk, just call Customer Relations at (800) 223-8720. Outside the US, please call (502) 491-1900.



Query Tip

Creating queries that select the top or bottom records

Version 2.0

Access's query system provides several tools for selecting records. You can use the Criteria row to specify criteria that records must meet. You can also open the Totals row and select records based on summary information from record groups.

However, there's always room for improvement. In Access 1.x, you can't select the top or bottom records. You can open the Totals row and use the Min or Max total, but the query will return only one record. You may want to see the top several records. Furthermore, because you must use the Totals row to select the first or last record, the resulting datasheet isn't updateable.

In this article, we'll show you an Access 2.0 feature that lets you select several records at the top or bottom of a record set. The query's datasheet is updateable, and Access returns the result very quickly.

Setting the Top Values property

To create a query that retrieves records from the top of a record set, you first design an ordinary select query. You include the tables in the Query window's table region and then drag the appropriate fields to the QBE grid. You also provide criteria where appropriate—just as you'd do for an ordinary select query. Finally, after you're sure the query is generating the correct set of records, you set the query's Top Values property.

Yes, that's right: the Top Values *property*. In Access 2.0, queries and their components have properties just as form and report elements do. To set the Top Values property, you open the property sheet by clicking the Properties button () in the tool bar. You then click in the table's query region to select the query's properties (as opposed to the field properties of the individual columns). Top Values is the third property in the list, as shown in Figure A.

Figure A

To select the top records, you set the query's Top Values property.

You type the number of top values you want to see into the Top Values property. For instance, if you want the records with the top five values, you assign *5* to the Top Values property. You can also choose Top Values by percentage. If you want to select the records in the top 5 percent of values, you assign *5%* to the property.

Keep in mind that the Top Values property lets you select the top or bottom records as they appear in the datasheet. It doesn't select maximum or minimum values.

Selecting the bottom values

You use the Top Values property to select values at the bottom of a record set as well. As you did for selecting top values, you simply specify the number or percentage of values you want to see. However, you must also reverse the sort order of the query. That way, the bottom records become the top records.

You reverse the sort order by using the Sort row of the QBE grid—just as you do in Access 1.x. As you probably know, you can set the Sort cell of any field to either *Ascending* or *Descending*. You leave the cell blank for fields that don't affect the sort order.

Often, you may not set the sort order of a query. If you don't, Access will list the selected records in a default order. We suggest you always define a sort order in Top Values queries—regardless of whether you want to see the top or bottom rows of a record set. That way, you'll always know what to expect.

Top values or top records?

Before we show you an example of a Top Values query, we'll clear up one point of possible confusion. By setting the query's Top Values property, you say you want to see all the records that are in the top *x* values or in the top *x* percent of values. You do not say you want the top *x* records or the top *x* percentage of records. What's the difference? Well, if every record had a different value in the sorting field, there wouldn't be a difference. However, several records might have the same values in the field by which you're sorting. If multiple

records share a top value, the datasheet will return more records than you expect.

When you think about it, Access must return the top values rather than the top records. Consider the following situation: Suppose you want to select the top five records of the table but two records have the fifth top value. Which of these two should the query select? The only good answer is to select both: The query must select the six records with the five top values.

An example

Let's look at a simple example. Suppose you want to give an award to the five employees who have the longest service with your company. In Access 2.0, you can create a query that selects the employee records with the first five hire dates. You create a query on your Employees table, sort the records by the Hire Date field, and assign *5* to the Top Values property.

We'll create our query on the Employees table of the NWIND.MDB database. Open the Access 2.0 version of the NWIND.MDB database file, highlight the Employees table in the Database window, and click the New Query button () in the tool bar.

In the New Query dialog box, click the New Query button. When the Select Query window appears, drag the asterisk from the field list to the QBE grid. Next, drag the Hire Date field to the grid and deselect the Show box. Then, move to the Hire Date column's Sort cell, click the dropdown arrow, and select *Ascending*. Your query should look like the one shown in Figure B.

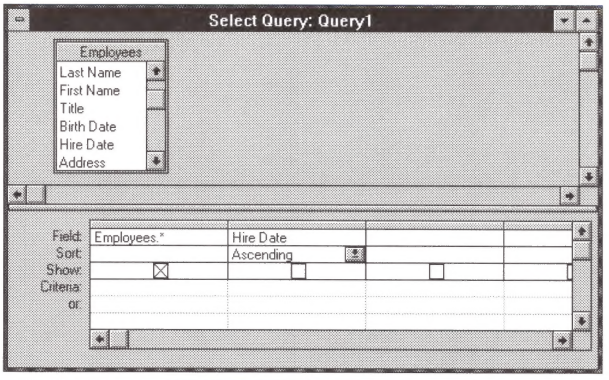
Your query will select all employees from the table, sorting them in ascending order by Hire Date. In other words, your long-time employees will appear at the top of the datasheet and your most recent hires will appear at the bottom. Click the Datasheet button () on the tool bar to see the results, as shown in Figure C.

To select only the first five employees, first click the Design View button () in the tool bar. Next, open the property sheet by clicking the Properties button. (If the property sheet's title bar doesn't say Query Properties, click in an empty place of the query's table region to select the query properties.) Next, move to the Top Values property and enter *5*. Now, click the Datasheet button to see the results shown in Figure D.

As you can see, the query selects six records rather than five! If you look closely, you'll notice the last two employee records in the datasheet have the same Hire Date—13-Sep-92. Remember, Top Values queries select the records with the top values—they don't select the top records. In this instance, you'll just have to honor six employees.

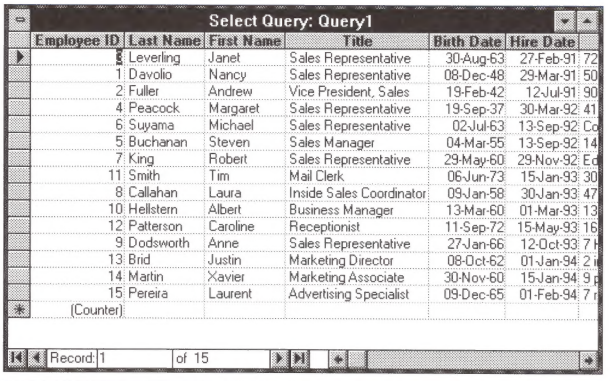
If, on the other hand, you want to identify the five most recent hires, you return to

Figure B



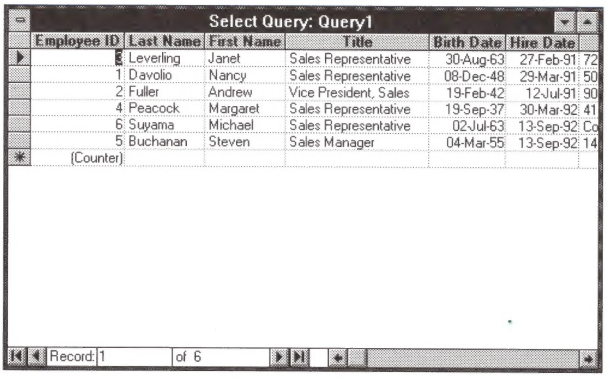
Your query should include all fields and sort on Hire Date.

Figure C



The select query will sort your employees by Hire Date.

Figure D



The query selects the employee records with the first five Hire Date values.

Design View and reverse the sort order. You do so by changing the Hire Date column's Sort cell from *Ascending* to *Descending*. That way, the query's datasheet will list employees from the most recent Hire Date (or largest date value) to the earliest Hire Date (smallest value). Then, the query's Top Values setting will tell the query to select the five most recent hires, as shown in Figure E.

Comparing Top Values queries to First or Last queries

Suppose you want to select the top or bottom value. As you'd expect, you set the query's Top Values property to 1. Now, let's compare such a Top Values query to a totals query that uses the First or Last operation to select a single record.

First, if you choose the Top Values query, the query's datasheet will be updateable. In other words, changes you make to the record selected by the query are permanent changes to the underlying table data. On the other hand, you create a totals query when you use the First and Last operators, so you can't update the First or Last query's datasheet.

Also, Top Values queries are usually much faster than totals queries. Our tests show that Top Values queries are often many times faster than similar queries that use the First or Last total operators to select the single record with the top values.

Figure E

Select Query: Query1

Employee ID	Last Name	First Name	Title	Birth Date	Hire Date
15	Pereira	Laurent	Advertising Specialist	09-Dec-65	01-Feb-94
14	Martin	Xavier	Marketing Associate	30-Nov-60	15-Jan-94
13	Brid	Justin	Marketing Director	08-Oct-62	01-Jan-94
9	Dodsworth	Anne	Sales Representative	27-Jan-66	12-Oct-93
12	Patterson	Caroline	Receptionist	11-Sep-72	15-May-93
(Counter)					

Record: 1 of 5

By reversing the sort order, you can select the most recent hires.

Conclusion

In this article, we introduced you to Access 2.0's Top Values query. You can now select the records with the top or bottom *x* values. You enter into the query's Top Values property the number of top values you want to see. Alternatively, you can provide a percentage value to see records with a top or bottom percentage of values. ❖

Letting Access dial the number

Several readers have called us about an error message they received when they implemented the telephone dialing technique described in February's "Letting Access Dial the Number." Although we believe several factors may be contributing to the problem, we've found two remedies that seem to correct the difficulties in all cases.

Let's first review the original technique. Basically, we showed you a batch file named DIALNUM.BAT that echoes Hayes-modem dial commands and a phone number to your computer's serial port. Your modem, which resides at the serial port, interprets the command and dials your number. You run the batch file from an Access form by using the Shell() function. You assign the function call to a command button's On Push property.

The Shell() function call we printed seems to be the problem. We recommended you assign the statement

```
=Shell("C:\ACCESS\DIALNUM ![Work Phone]!")
```

to the OnPush property. If this statement causes problems, try adding the .BAT file extension to the DIALNUM filename so that the call reads

```
=Shell("C:\ACCESS\DIALNUM.BAT ![Work Phone]!")
```

If that doesn't work, try stripping out the path so that the function call becomes

```
=Shell("DIALNUM.BAT ![Work Phone]!")
```

Everyone we've spoken to has solved the problems by making one or both of these changes. If you continue to have problems after trying these workarounds, please call us at (800) 223-8720.

Combining several tables into a single record set

Version 2.0

We maintain a central listing of our company's customer data, compiling the list from about 20 field offices. To pull the data together, we generated a new table and then used append queries to merge the many tables into the new one. Is there a better way to do this? Can I combine the tables with a query?

John Strang
Fort Lauderdale, Florida

Mr. Strang will be happy to hear that Access 2.0 supports SQL union queries, which do exactly what he's asking for. Union queries combine multiple tables into one datasheet.

However, creating a union query is much different than creating other types of queries. Rather than using the QBE grid, you must provide the SQL statement that defines the query. If you've never written a SQL statement, you may be reluctant to start—especially if you're just getting the hang of the QBE grid. Fortunately, you can express any query you can create with the QBE grid. You can start by creating an ordinary select query with the QBE grid and then let Access generate your query's SQL code. We'll first show you a SQL statement that creates a union query. We'll then show you an easy way to create the query.

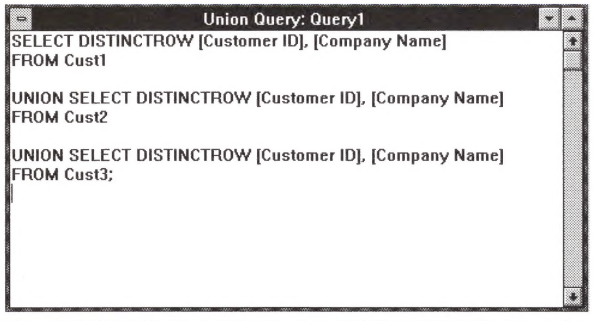
An example

For the purposes of our example, let's assume you need to merge the three tables Cust1, Cust2, and Cust3. The SQL query shown in Figure A selects the Customer ID and Company Name fields from the three tables and lists them in one record set.

Notice that the three components of this SQL query are very similar. The query starts with a SELECT statement, which selects the Customer ID and Customer Name fields from the Cust1 table. It then includes nearly identical SELECT statements for the Cust2 and Cust3 tables. The only difference between the first SELECT statement and the other two is the UNION keyword. Here, UNION is the keyword that tells the query to combine the records from that table with the records of the previous SELECT statement's table.

Before we show you the steps for creating this query, we'll briefly describe the meaning of the SQL statement's components. You use the SELECT keyword to specify the fields the query will include. Including a field's name in the SELECT command is equivalent to creating a column in the QBE grid. You use the FROM keyword to specify the table from which you'll draw the data. There are many other keywords you can use in creating a SQL query. We'll discuss other SQL tips in future articles.

Figure A



This query selects the records from the Cust1, Cust2, and Cust3 tables.

Creating the union query

To create this query, you first click the Query button in the Database window and then click the New button. When the Add Table dialog box appears, click the Close button without selecting any tables. Then, pull down the Query menu and click the SQL Specific selection. In the submenu that appears, click Union. Access will display a blank Query window. You then type the SQL code shown in Figure A. When you run the query, the records from all three tables will appear in the datasheet.

There are a few tricks you can use to create the union query. When you create one for your specific database, you may want to use the following shortcuts:

- Generate the initial SELECT statement by first building the query with the QBE grid and then

Microsoft Access
Technical Support
(206) 635-7050

Please include account number from label with any correspondence.

issuing the SQL command on the View menu.

- Build the additional components of the query by first generating the SQL code from the QBE grid query and then inserting the UNION keyword before the SELECT keyword.

- Remove the semicolon from the end of all components except the last. In other words, a semicolon must end the query but must not appear elsewhere in the query definition.

Notes

Although union queries do a marvelous job of pulling together the data from multiple tables, they don't produce an updateable datasheet. You can view the data from the union query's datasheet, but you must make changes in the original tables.

Also, note that you can combine tables even when the tables' fields have different names. Access expects you to enter the same number of fields in each component's SELECT statement and expects them to have the same data types. The query will match up the fields in each position. The query will use the field names you list in the first SELECT statement. ❖

Covering Access 2.0 in *Inside Microsoft Access*

As you read in last month's issue, Microsoft has released Access 2.0. As you decide whether to upgrade—and whether to continue your subscription to our journal—you're probably curious how *Inside Microsoft Access* will change with the new version.

Well, one look at Access 2.0 told us that the vast majority of you will eventually upgrade. To keep pace, we'll upgrade as well. However, we recognize the decision to upgrade doesn't occur overnight, and the software never seems to arrive fast enough either. Consequently, we'll begin including Version 2.0-specific articles, but we'll continue to provide Access 1.x tips and techniques for several months. The article on page 12, "Creating Queries that Select the Top or Bottom Records," will give you a good sample of the coverage you can expect over the next few months.

Furthermore, if you subscribe to our Access Resource Disk, you'll receive database files for all three versions of Access—1.0, 1.1, and 2.0. Of course, we won't include the objects for the Version 2.0-specific articles in the 1.x databases. But Access Resource Disk subscribers will continue to receive all the data and objects we describe in the journal. We know you're eager to begin taking advantage of Access 2.0's new features. We look forward to supporting you through the transition.

Printed in the USA
This journal is printed on recycled and recyclable paper.
50% recycled fiber including 10% post-consumer waste